Enterprise Information Systems
Vol. 00, No. 00, Month 200x, 1–24

The business process management literature describes a multitude of approaches (e.g. imperative, declarative or event-driven) that each result in a different mix of process flexibility, compliance, effectiveness and efficiency. While the use of a single approach over the process lifecycle is often assumed, transitions between approaches at different phases in the process lifecycle may also be considered. This paper explores several business process strategies by analyzing the approaches at different phases in the process lifecycle as well as the various transitions.

Keywords: Business Process Management; Process Modeling; Process Enactment; Process Transitions, Process-Aware Information Systems

ISSN: 1751-7575 print/ISSN 1751-7583 online

DOI: 10.1080/1751757YYxxxxxxxx
http://www.informaworld.com

Enterprise Information Systems
Vol. 00, No. 00, Month 200x, 2–24

RESEARCH ARTICLE

Exploring Business Process Modeling Paradigms and Design-Time to Run-Time Transitions

(*Received 00 Month 200x; final version received 00 Month 200x*)

Introduction

Organizations face a continuous pressure to improve process compliance, flexibility, efficiency and effectiveness. While responding to these pressures individually can be demanding, the real challenge is dealing with the intrinsic tradeoffs. For example, process compliance requires control over processes, whereas Regev suggested that process flexibility is desirable to be able to support non-standard cases (Regev et al. 2006, Fan et al. 2000). The business process management literature describes multiple approaches – including the imperative e.g. (Zisman 1977) and the declarative e.g. (van der Aalst et al. 2009b) business process modeling paradigm as well as a series of hybrid paradigms e.g. (Hallerbach et al. 2010b, Lu et al. 2009, Sadiq et al. 2005) – which propose different sets of tradeoffs.

While a diversity of approaches has been covered by the literature, most contributions consider the selection of the optimal approach given the business environment as a one-time choice at the process design phase. However, business processes may also require that different tradeoffs are made at different phases of the process lifecycle. The contribution of this position paper will be the exploration of various business process strategies that combine the selection of a design-time paradigm and a run-time paradigm (i.e. position selection) with a transition path. These strategies may result in a better fit between the business processes, the business environment and the systems that support them. Rather than making a value judgement of the different strategies, we will focus on a discussion of the impact on the process characteristics.

The remainder of the paper is structured as follows. In the next section the business process modeling paradigms are categorized and described. The subsequent sections define and analyze the different positions in both layers. The design-time to run-time transitions are uncovered and discussed in-depth with special attention for cross-paradigm transitions and specific use cases. The final section concludes the paper.

Business Process Modeling Paradigms

In the business process management literature a wide spectrum of business process modeling paradigms has been presented. At the extremes of this spectrum we find the imperative and the declarative process management paradigm. In between there is a wide variety of hybrid paradigms that combine aspects of both extremes. The different paradigms can be roughly categorized into the following classes:

- The **imperative business process modeling paradigm** focuses on defining an activity sequence that will result in obtaining the related corporate goal. These activity sequences can be easily represented in a graph-based language. Typical use cases include the processing of static and standardized financial transactions. Representative contributions for this paradigm include (Zisman 1977, Object Management Group 2006, Ellis and Nutt 1993).
- The **declarative business process modeling paradigm** focuses on capturing and defining regulatory or internal directives in constraints, rules, event conditions or other (logical) expressions, e.g. sequence relationships among the different activities. With a minimum specification of the relevant business concerns, some freedom is left for the exact activity sequence for each process instance. Declarative business process strategies are often used for business processes that can be characterized as dynamic and non-standard, such as health care processes. The declarative approach to business process management is described in e.g. (Goedertier and Vanthienen 2009, van der Aalst et al. 2009b, Reichert and Weber 2012). Recently, the case management approach is receiving growing attention in the declarative process literature e.g. (Swenson and Palmer 2010, Martens et al. 2011).
- **Hybrid business process modeling paradigms** focus on combining activity sequence specifications with declarative specifications, the principles of the other two business process modeling paradigms. Multiple business process types and business environments may require hybrid business process strategies, e.g. business processes which are partially static and partially dynamic such as consulting processes. Examples of scientific contributions that propose hybrid business process strategies include (Sadiq et al. 2005, van der Aalst et al. 2009a, Schonenberg et al. 2008a, Kumar and Yao 2009, Hallerbach et al. 2010a, Sinur 2009, Doehring et al. 2010).

The following sections will analyze the business process modeling paradigms at different phases in the process lifecycle, and will look into possible strategy combinations during that lifecycle.

The Different Design-Time and Run-Time Positions

The traditional business process lifecycle consists of four phases with distinct roles (Weske 2007). During the *process design phase* a specific part (e.g. order-to-cash or input procurement) of the business strategy is specified in a process model. The *process implementation phase* deals with configuring the selected process execution environment (i.e. the business process management system). At *process enactment* the individual process instances are executed. Finally, the information gathered during the process enactment phase will be thoroughly analyzed during the *process evaluation phase* for audit, business

process reengineering or other purposes. While the process **design-time** obviously coincides with the process design phase, the process **run-time** consists of both the process implementation and enactment phase (figure 1). As the process analysis and evaluation phase often results in specific recommendations/requirements for a process redesign, it can be considered as part of the next cycle's design-time.

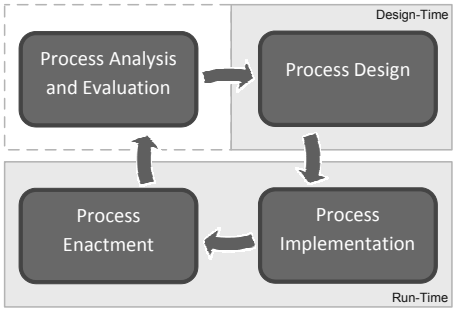


Figure 1.: Business process lifecycle

Different frameworks have described a detailed roadmap for the design-time process model development and focus on taking into account multiple aspects of business processes (work coordination, resource allocation, strategic decisions and involved data). The Architecture of Integrated Information Systems approach (ARIS, (Scheer 2000)) develops a four stage roadmap; starting with establishing the initial strategic situation over a requirements engineering phase and a design specification phase to the final implementation description. Additionally, the ARIS framework focuses on an integrated approach to application system development by taking into account different process views, i.e. the function view (goals and objectives), the organization view (resources), the data view (data process environment and triggers), the output view (physical and non-physical input and output) and the control view (description of the relationships).

In (Karagiannis et al. 1996), Karagiannis presents the development of a business process management system (BPMS) as a process in itself, which consists of five subprocesses. This approach also provides a thorough division of the design phase in multiple subphases, which are called subprocesses. The approach starts with the making of a strategic decision based on the objectives and constraints for a selected set of business processes. In the reengineering subprocess a detailed insight of the selected processes is acquired, followed by the resource allocation subprocess that analyses the resource aspect. The last two subprocesses, i.e. the workflow management process and the performance evaluation subprocess, coincided with respectively the run-time and the process analysis and evaluation phase.

Oesterle developed a three layer based business engineering approach to describe the corporate reality (Österle 1995). The strategy layer deals with the specification of the business model and proposes the corporate goals and objectives, whereas the output required to fulfill these goals is created in the process layer. Finally, the information systems layer is focused on supporting the processes in the creation of the outputs.

In the remainder of this section we characterize the design-time and run-time positions, which may be the imperative, declarative and hybrid business process modeling paradigms at design-time and at run-time, respectively.

Design-Time Positions

In this subsection the design-time modeling approaches provided by each of the business process modeling paradigms are described. Each business process modeling paradigm specifies a different set of design principles and offers a different set of constructs to model business processes.

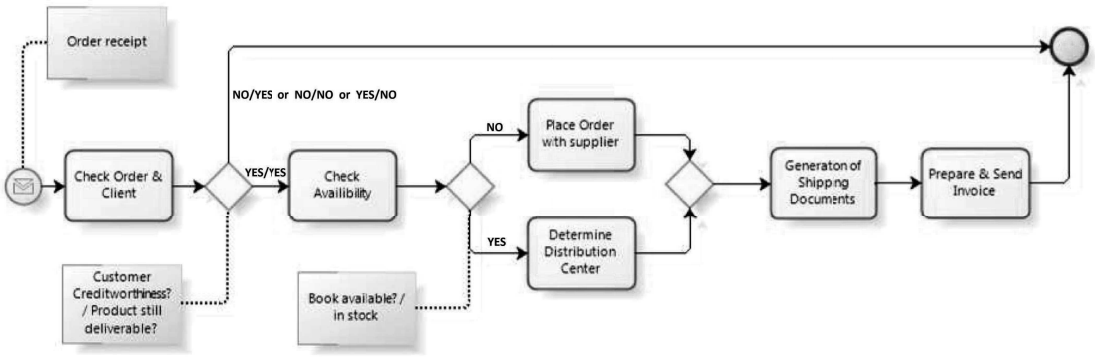


Figure 2.: Imperative process for a typical order-to-cash process of an online shop (in BPMN)

Acquiring business process models is an arduous and time consuming activity (Herbst and Karagiannis 1999). The process knowledge is usually distributed in the heads of multiple actors, who are directly involved in the execution of the real-life business process. Analysts and modeling experts typically use an iterative approach based on amongst others interviews and observations, to come to a formal description the business process. As described in (Curtis et al. 1992), a business process consists of multiple aspects; the functional, control flow, organizational and data perspective. The workflow is a formal specification of the control flow of a business process (Karagiannis et al. 1996) and is the perspective on which most of the contemporary research focuses (Chiao et al. 2013). Recently, the research area of process mining, i.e. collection of machine learning approaches to induce a workflow model from the event log, has gained significant attention (Herbst and Karagiannis 1999, van der Aalst et al. 2007, Van der Aalst 2011). The starting point in such a business process modeling exercise is the as-is process situation, which is in sharp contrast with the development of a to-be process from scratch.

Imperative process models contain a precise definition of the control-flow of the business process in a graph-based process modeling language. The basic constructs of graph-based process modeling languages are activities and control-flow dependencies between them, represented as nodes and directed arcs respectively. Several graph-based process modeling languages offer a set of additional constructs, e.g. events, data objects or compensation associations. A multitude of graph-based process modeling languages have been presented; among others Petri Net (based) modeling (Zisman 1977, Ellis and Nutt 1993), IDEF3 (Mayer et al. 1995), BPMN (Object Management Group 2006), UML Activity Diagram (Object Management Group 2004) and EPC (Keller et al. 1992). An example can be found in figure 2.

Declarative process models focus on what should be done in order to achieve business goals, not how it should be done. Therefore, these models specify a set of constraints, business rules, event conditions or other (logical) expressions that define properties of

and dependencies between activities in a business process. Consequently, all alternative paths are implicitly specified and defined as the paths that do not violate the business rules. A wide variety of declarative modeling approaches has been specified in business process management, from basic ECA-rules (Kappel et al. 1998) (including the ECA-based CIMOSA declarative rules (Vemadat 1998, ESPRIT Consortium AMICE 1993) to the declarative process modeling languages ConDec (Pesic and van der Aalst 2006), DecSerFlow (van der Aalst and Pesic 2006) and BPCN (Lu et al. 2009). In (Goedertier and Vanthienen 2009) we presented a comprehensive review of the most common declarative process modeling languages. An example can be found in figure 3.

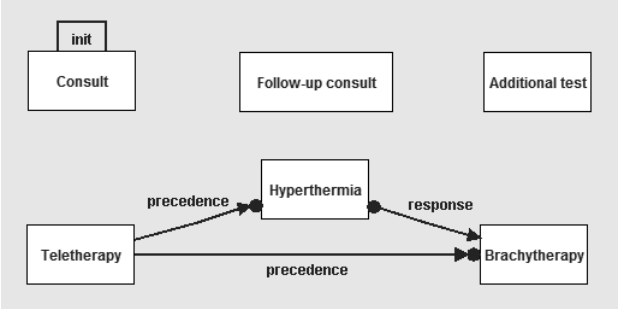


Figure 3.: Declarative process for a radiotherapy process (guidelines based on previous process mining research (Caron et al. 2012), in ConDec)

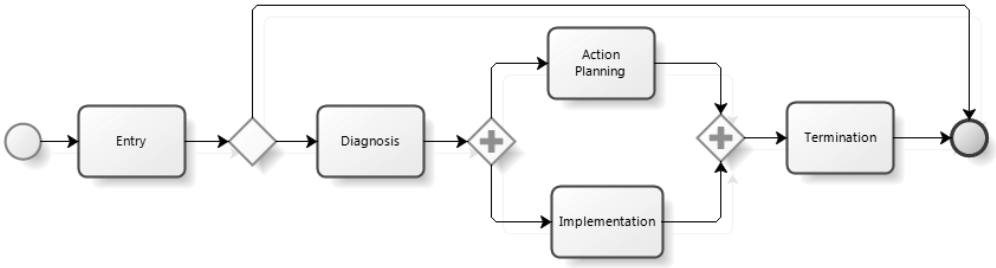
Hybrid process models combine both imperative and declarative modeling constructs to specify the process. Several design-time modeling approaches have been presented in the context of hybrid process strategies, however, most of them can be classified into two categories.

- *Process models with placeholder activities* form a type of process models specified in either an imperative or declarative process modeling language. The placeholder activities encapsulate subprocesses that are defined in a process modeling language of the other business process modeling paradigm, e.g. (Sadiq et al. 2005, van der Aalst et al. 2009a, Schonenberg et al. 2008a). An example can be found in figure 4.
- *Imperative process models with rule-based adaption* are hybrid process models that use business rules (i.e. declarative specification) to adapt an imperative reference model to specific the specific needs of individual business cases, e.g. (Kumar and Yao 2009, Hallerbach et al. 2010a). An example can be found in figure 5.

Run-Time Positions

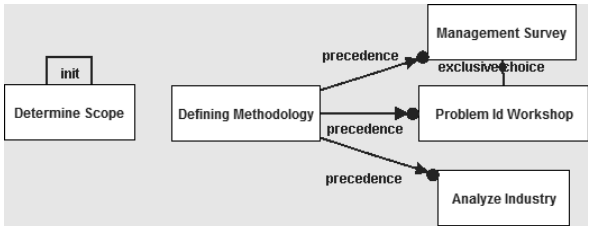
Also at run-time a strong differentiation can be observed between the three business process modeling paradigms. This section briefly describes the execution principles of each business process modeling paradigm.

Imperative process enactment techniques allow for automatic routing of the work based on exactly specified execution paths. Consequently, imperative process enactment requires a full specification of each valid execution path in the process model (Aguiar and Weston 1995, Bussler 1996, Burstein et al. 2005). Examples of imperative process



(a) Overall consulting process

(b) Imperative 'entry' sub process (in BPMN)



(c) Declarative 'diagnosis' sub process (in ConDec)

Figure 4.: Hybrid process model for a typical consulting process

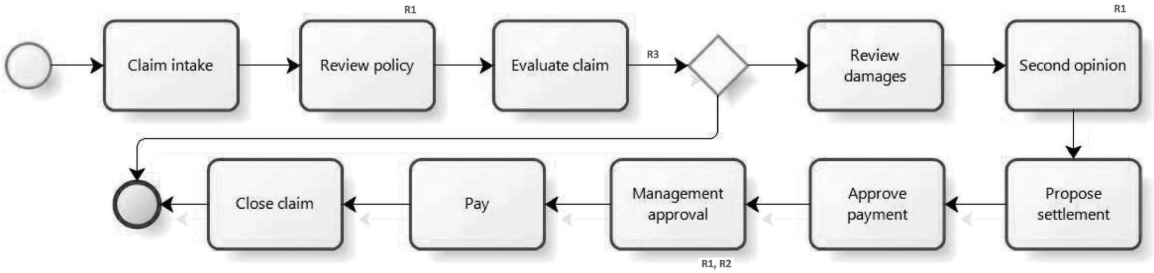
execution languages are BPEL (Organization for the Advancement of Structured Information Standards (OASIS) 2007) (whether or not extended to BPEL4People) and YAWL (van der Aalst and Ter Hofstede 2005).

Declarative process enactment techniques support a dynamic run-time development of the execution scenario of an individual process instance. State-of-the-art declarative process-aware information systems often support constraints which are specified in specific types of logic, e.g. LTL (Pesic et al. 2008) and CTL (Yu et al. 2006). Different approaches for the actual enactment of declarative process models have been proposed, including the use of dynamic planning algorithms (e.g. PLMflow) (Zeng et al. 2002, Barba et al. 2012, Chua et al. 2013, Qin and Fahringer 2012), of ECA rules (Kappel et al. 1998), of event-driven BPM approaches (Paschke and Boley 2009) and the construction of LTL-automata (Pesic et al. 2008).

Hybrid process enactment techniques are used to enact process models that contain placeholder activities. The base process is executed in accordance with the execution principles of the base paradigm, while placeholder activities are enacted by the principles of the other paradigm. Two approaches to switch enactment strategies have been proposed; the use of build-activities executed in the same workflow engine (e.g. Chameleon) (Sadiq et al. 2005) and the use of subprocesses encapsulated in a service (van der Aalst et al. 2009a, Reichert and Weber 2012).

Detailed Analysis of the Design-Time and Run-Time Positions

Since every position has its own design or execution principles, all positions have different characteristics and consequently appeal to different business requirements. In (Davenport 1993) Davenport identifies four desirable qualities for business processes; process flexi-



(a) Imperative reference model (in BPMN)

R1: If claimed amount < 1000 then skip review policy, skip second opinion and skip management approval.
R2: If claimed amount > 1000000 then management approval is needed
R3: If insurance type = fire-insurance then insert give an advance after evaluate claim.

etc.

(b) Configuration or materialization rules

Figure 5.: Hybrid process model for a typical claim handling process (adaption of process in (Kumar and Yao 2009))

bility, compliance, effectiveness and efficiency. This section analyzes the possible impact of each position's principles on these characteristics.

- **Process flexibility** is the extent to which an organization can deal with business process change, the ability to accommodate the special needs of particular business process instances as well as to accommodate process model evolutions.
- **Process compliance** is the extent to which a process is in correspondence with the internally defined business rules and the externally imposed business regulations.
- **Process effectiveness** is the extent to which a business process realizes its business goals.
- **Process efficiency** is the extent to which the organization of the business process is capable of minimizing the amount of utilized resources such as personnel, materials, time, machine capacity.

Of the aforementioned performance criteria, efficiency and effectiveness are the most crucial for operational excellence, yet to some extent compromised by flexibility and compliance.

Design-Time Positions

Based on the aforementioned desirable qualities, this paragraph evaluates the different business process modeling paradigms at design-time. Additionally, two important characteristics of process modeling languages are taken into account:

- The **expressibility** of a process modeling language is determined by its ability to

express specific process elements, e.g. control-flow, data, execution and temporal information (Lu and Sadiq 2007, zur Muehlen et al. 2007, Recker 2010)

- The level of **comprehensibility** reflects the ability of a process modeling language to define understandable process models that can be easily communicated among various stakeholders (Fahland et al. 2009a, Bandara et al. 2005).

Imperative Process Modeling results in an exact specification of all the alternative execution paths, events and exceptions. These specifications can be easily represented in a graph-based language. However, control-flow dependencies that are not dictated by internal or external directives may be modeled, i.e. *overspecification* of the workflow model. To overcome the impact of overspecification on the built-in flexibility, several approaches have been proposed. Firstly, the regulation of the granularity of a process activity allows for more flexibility by excluding the dynamic aspects from process control (Sadiq et al. 2001, Weber et al. 2009). Secondly, flexibility by design also known as flexibility by enumeration or advanced modeling, stimulates the process designer to model different known execution paths (Regev et al. 2006). Finally, process variants can be derived from a reference model to support different execution scenarios (Rosemann and van der Aalst 2007).

Imperative process models that are based on *formal semantics* and contain precise specifications, process compliance can be efficiently checked (van der Aalst 1997). However, by implicitly modeling the business concerns in control-flow dependencies business logic and rules are frequently duplicated, which might result in *maintainability issues* (Fahland et al. 2009b). Changes in process compliance requirements or the business environment can trigger a series of process evaluation and reengineering activities (Goedertier and Vanthienen 2009). The impact of an imperative process model on the actual business process's efficiency and effectiveness directly depend on the skills of the process designer.

Imperative process modeling languages commonly provide a clear visual representation, which generally leads to *comprehensible process models* if they are not too large. An example can be found in figure 2.

Declarative Process Modeling literature is characterized by its aim to *reconcile process flexibility and compliance*. Adapting the business rules to changes in the business policies or imposed regulations is straightforward, since there is no duplication of business logic in a declarative process model (nor over multiple process models). Moreover, additional business rules can be directly added, without the need to fully redesign the business process. Consequently the design-time flexibility of a declarative business process model can be high.

Process compliance might be fully guaranteed when all relevant business policies and regulations are mapped on mandatory business rules, which results in traceability and facilitates verification by domain specialists. In order not to affect the process effectiveness, constraints should be valid, consistent, feasible etc. The impact on the process efficiency of the declarative process model can be influenced by specifying guideline constraints, which specify an optimal execution path but can be violated during execution. An example can be found in figure 3.

Languages representing rule-based process modeling provide a higher *expressibility* than graph-based languages (e.g. the ability to specify temporal requirements) (Lu and Sadiq 2007), but might result in process models which are *less comprehensible* (Fickas 1989) due to large and possibly unstructured sets of business rules.

Hybrid Process Modeling combines modeling principles of the imperative and the declarative process paradigm, which results in a *moderation of the impact* that the base model's business process modeling paradigm has on the process characteristics. Using declarative subprocesses in placeholder activities of an imperative process model (e.g. figure 4a) improves the impact on the process flexibility and reduce the effort to keep the declaratively specified subprocesses compliant with a dynamic business environment. On the other hand, the use of imperative subprocesses in placeholder activities of declarative process models seems to trade an improvement of the impact on the process efficiency for a decrease in the process flexibility.

By providing rule-based adaption abilities (example process in figure 5) the hybrid business process modeling paradigm is able to improve the process flexibility compared to the imperative business process modeling paradigm, i.e. the native paradigm of the reference model. The quality of the adaption rules can influence the impact of the derived process models on the process compliance, effectiveness and efficiency. Important questions in this context: Are all situations covered by the rules? Are there any contradicting rules? Will the business processes remain compliant after applying the customizing business rules?

Run-Time Positions

Comparable to the selection of a design-time position, the selection of a run-time position might influence the extent to which the optimal business process characteristics (dictated by the environment, such as efficiency) will be reached.

Imperative Process Enactment is characterized by a straightforward execution based on precisely specified execution paths. As a consequence imperative process enactment might result in considerable *process efficiency for static and standardized business processes* (Goedertier and Vanthienen 2009). On the other hand, due to these detailed specifications, *process flexibility remains limited*.

The business process literature, however, has proposed two sets of techniques to partially meet the needs of dynamic business environments. Firstly, flexibility by deviation methods that allow a run-time (partial) modification of the execution paths, e.g. case-handling (van der Aalst et al. 2005) and ADEPTflex (Reichert and Dadam 1998). Related to these techniques are the exception-handling approaches, e.g. the Exlet approach (Adams et al. 2007). Secondly, flexibility by change techniques that support structural adaptations of the execution paths at run-time (Weber et al. 2008, Bhat and Deshmukh 2005). Additionally, the dynamic binding of services to the business process can further introduce flexibility (Hrgovcic et al. 2011).

Process compliance and effectiveness are generally determined by the execution path specifications. However, the use of flexibility enhancing techniques might require compliance checking techniques (e.g. (Li et al. 2008, Ghose and Koliadis 2007, Hur et al. 2003)) to assure that the process execution remains in accordance with the business policies and the imposed regulation.

Declarative Process Enactment guarantees a *high run-time flexibility* for declarative process specifications that contain only the strictly required mandatory constraints. An individual execution path that satisfies the set of mandatory constraints can be dynam-

ically build for a specific process instance. Process *compliance is assured* when all regulation and business policies are correctly mapped onto mandatory business constraints.

During the construction of a suitable execution path *little support* is provided to the end user (Weber et al. 2009), which could affect the process effectiveness. In (Schmidt and Simonee 1996, Barba and Del Valle 2011) the idea of differentiating constraints by modality was proposed, soft constraints would guide the user whereas mandatory constraints would ensure compliant behavior. The guidance provided by the soft constraints might depend on explicit domain knowledge or can be learned through process mining (Schonenberg et al. 2008b). This guidance may result in an improved efficiency and effectiveness. Lastly, the increased size and complexity of contemporary process models might decrease the potential for process automation since it has been reported that current declarative workflow management systems might have *limited efficiency* in these cases (van der Aalst et al. 2009b).

Hybrid Process Enactment techniques execute the base process in accordance with the chosen process paradigm, whenever a placeholder activity is encountered a switch in process paradigm takes place. In general, the *qualities of the process parts are determined by the paradigm in which they were defined*. When opted for a 'flexibility as a service' solution (i.e. the placeholder activity is linked to a service) *additional process flexibility* can be realized as it becomes possible to select an optimal service out of an extensive repertoire (Papazoglou and Georgakopoulos 2003). However, the Ripple Down Rules used for service selection (van der Aalst et al. 2009a), should be checked for compliance with the different business concerns.

Design-Time to Run-Time Transitions

Traditional business process management solutions are oriented towards a single process paradigm, e.g. the business processes are (partially) modeled using imperative modeling languages (such as BPMN (Object Management Group 2006)) and then executed in an imperative enactment environment (such as BPEL (Organization for the Advancement of Structured Information Standards (OASIS) 2007)). These design-time to run-time transitions within a single process paradigm are the *same paradigm transitions*. While these traditional solutions allow to fully exploit the advantages of a single process paradigm, there exist business processes that have different requirements at design-time and run-time, e.g. in terms of flexilibity. In these cases we suggest and analyze design-time to run-time transitions between process paradigms, the *cross paradigm transitions*.

Three cross-paradigm transitions have been identified as highly interesting transitions. Both the transition from hybrid modeling to imperative enactment and from declarative modeling to imperative enactment are characterized by a need for design-time flexibility and run-time efficiency. The transition from imperative modeling to declarative enactment is motivated by the need for a distributed implementation at run-time combined with a clean process overview at design-time. Additionally, the transitions between paradigms at design-time or at run-time can be interesting; e.g. while declarative models can be easily maintained, imperative models allow a better communication with stakeholders. The remaining three potential transitions, appear to be less interesting as they contain only a partial solution, compared to the other three cross-paradigm transitions.

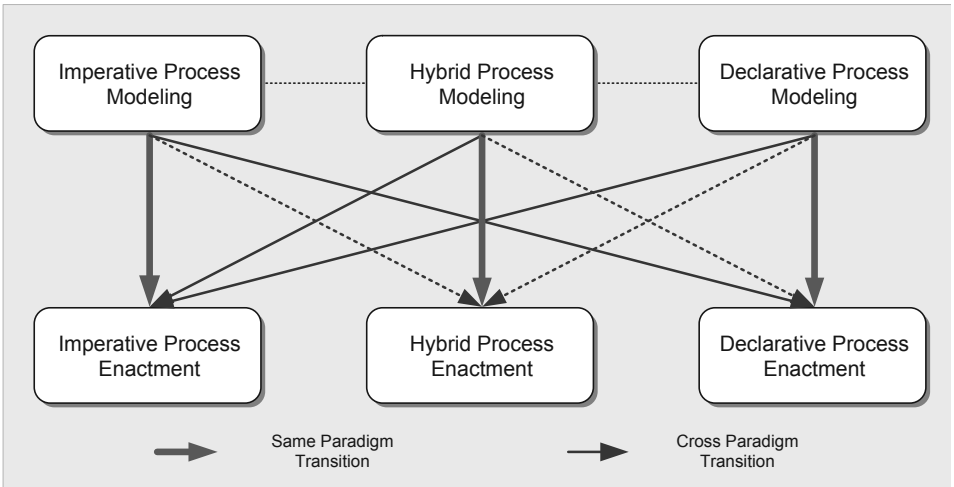


Figure 6.: Design-time to run-time transitions

The next section provides a detailed discussion of the different design-time to run-time transitions.

Detailed Description and Discussion of the Design-Time to Run-Time Transitions

The different design-time to run-time transitions will be further described in this section. Additionally, the impact of each transition type on the desirable business process characteristics will be analyzed and a business process example for each type will be provided.

Same Paradigm Transitions

In this subsection the same paradigm transitions, or design-time to run-time transitions within the same business process modeling paradigm, will be further discussed and evaluated.

Imperative - Imperative Transition. Different transformation strategies between imperative process modeling languages and imperative process execution languages have been proposed in the literature (Decker et al. 2008, Ouyang et al. 2009). However, due to a *conceptual mismatch* between the standard imperative process modeling languages and BPEL (Organization for the Advancement of Structured Information Standards (OASIS) 2007) (i.e. BPMN (Object Management Group 2006) and UML AD (Object Management Group 2004) provide a richer set of constructs than BPEL does), translation techniques are only offered for process models captured in a core subset of the imperative process modeling languages (Recker and Mendling 2006). Consequently, only for business processes modeled in the core subset, as defined by the chosen transformation strategy, it can be guaranteed that the process characteristics are not affected in this transition. The conceptual mismatch between standard imperative process modeling languages and YAWL (van der Aalst and Ter Hofstede 2005) is not as significant (Decker et al. 2008).

Impact on process characteristics. A set of research contributions on *process variant management* falls into the scope of the imperative-imperative transition, e.g. (Lu et al. 2009). The relevant process variant management approaches all query a repository of imperative process variants before run-time. The process variant that best fits the particular context is instantiated in the imperative execution environment. While this imperative-imperative transition might have a positive impact on the process flexibility, managing changing compliance requirements can be challenging for repositories with large collections of process variants. The impact on process efficiency and effectiveness depends on the quality of the process variants and the number of different situations captured by the process variants in the repository.

Example process case. The imperative-imperative transition can be recommended for business processes in a *stable environment with predictable execution paths*. Under these conditions optimal process efficiency can be guaranteed. This transition type is typically used for processing standard and static items, such as online orders or standardized financial transactions. Listing 1 provides the BPEL run-time process model for the order-to-cash process in figure 2.

Listing 1: Imperative-Imperative Transition: BPEL run-time model for BPMN design-time model (figure 2)

```
<process xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
    >
...
   <sequence>
      <invoke name="CheckOrderAndClient" />
         <switch>
            <case condition="order and client are acceptable">
               <invoke name="CheckAvailability" />
                  <switch>
                     <case condition="available">
                        <invoke name="PlaceOrderWithSupplier" />
                     </case>
                     <case condition="not available">
                        <invoke name="DetermineDistributionCenter" />
                     </case>
                  </switch>
               <invoke name="GenerationOfShippingDocuments" />
               <invoke name="PrepareAndSendInvoice" />
            </case>
            <case condition="order or client or both are not acceptable">
            </case>
         </switch>
   </sequence>
</process>
```

Declarative - Declarative Transition. The common declarative process modeling languages have their roots in formal logic. Due to this formal foundation the translation of a high-level declarative process model into enactable rules is *rather straightforward* (e.g. translation of a ConDec process model into LTL expressions (Pesic et al. 2008)). These enactable rules are then used to govern declarative process instances by non-deterministic workflow engines (i.e. rule-based event-driven process engine).

Impact on process characteristics. During the declarative-declarative transition no (additional) factors that affect the business process's flexibility, compliance, efficiency or effectiveness will/can be introduced.

Example process case. Declarative-declarative transitions are suitable for business processes in a *highly evolving environment* and/or business processes with *non-predictable*

execution paths. Process compliance is guaranteed when all regulation and business policies are correctly mapped onto mandatory business constraints. As declarative process management systems might provide limited support at run-time (Weber et al. 2009), this transition type will be most suited for experts dealing with unique cases (Schmidt 2006). Currently, case management is gaining increased attention in this segment (Swenson and Palmer 2010). Typical use cases for the declarative-declarative transition include the processes that deal with unique cases and as a consequence require flexibility, such as non-standardized health care processes. Listing 2 contains the LTL-based process model for the gynecologic oncology process in figure 3. This code can be directly used for enacting the declarative process model in the Declare business process management system (Pesic et al. 2007).

Listing 2: Declarative-Declarative Transition: LTL run-time model for ConDec design-time model (figure 3)

```
1 ((Consult,start) \/ (consult,complete)) W ((Follow-up,start) \/ (Additional
      test,start) \/ (Teletherapy,start) \/ (Hyperthermia,start) \/ (
      Brachytherapy,start))     \\ initial(Consult)
2 !(Hyperthermia,start) W (Teletherapy,complete)     \\ precedence(Teletherapy,
      Hyperthermia)
3 !(Brachytherapy,start) W (Teletherapy,complete)     \\ precedence(Teletherapy,
      Brachytherapy)
4 [](((Hyperthermia,complete) => <> (Brachytherapy,start))     \\ response(
      Hyperthermia,Brachytherapy)
```

Hybrid - Hybrid Transition. The hybrid-hybrid transition can only be applied on hybrid process models that contain placeholder activities. Within the context of this type of hybrid business process modeling paradigm, *the business process modeling paradigms of each process parts determine which same paradigm transition will be used for that process part*. An imperative-imperative transition will be observed for the overall business process if the core process is specified in an imperative process model, while for the subprocesses contained in the placeholder activities a declarative-declarative transition will be noted.

Impact on process characteristics. As these transitions are intrinsically similar to the previously described same strategy transitions, we argue that the impact on the process characteristics of this transition is determined by those same strategy transitions. It should be noted that process variant management approaches in this context will be easier to maintain than those presented in the context of imperative-imperative transitions, since the base structure is not duplicated.

Example process case. Hybrid-hybrid transitions will be used for business processes that contain *both process parts with stable and highly evolving environments* and/or that consist of *both process parts with predictable and non-predictable execution paths*. A typical case for the hybrid-hybrid transition is a process that supports an advisory project of a consulting company. These processes combine very structured process parts, such as administrative activities at the beginning and the end, with an unstructured set of problem-solving activities in the middle. Listings 3 to 5 present the run-time models for the different process fragments of the consulting process model in 4.

Listing 3: Hybrid-Hybrid Transition: Run-time BPEL model for the overall consulting process (figure 4a)

```
1 <process xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
      >
2 ...
3    <sequence>
4       <invoke name="Entry" />
```

```
5       <switch>
6          <case condition="Acceptable">
7             <sequence>
8                <invoke name="Diagnosis" />
9                <flow>
10                  <invoke name="ActionPlanning" />
11                  <invoke name="Implementation" />
12               </flow>
13               <invoke name="Termination" />
14            </sequence>
15         </case>
16         <case condition="NotAcceptable">
17         </case>
18      </switch>
19   </sequence>
20 </process>
```

Listing 4: Hybrid-Hybrid Transition: BPEL run-time model for the Entry subprocess (figure 4b)

```
1 <process xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
      >
2 ...
3    <sequence>
4       <invoke name="CollectOrientationData" />
5       <invoke name="InvestigationalInterviews" />
6       <invoke name="PreliminaryProblemAnalysis" />
7       <invoke name="DraftTermsOfReferenceAndProposal" />
8       <invoke name="NegotiateConsultingContract" />
9    </sequence>
10 </process>
```

Listing 5: Hybrid-Hybrid Transition: ConDec run-time model for the diagnosis subprocess (figure 4c)

```
1 ((DetermineScope,start) \/ (DetermineScope,complete)) W ((DefiningMethodology,
      start) \/ (ManagementSurvey,start) \/ (ProblemIdentificationWorkshop,start
      ) \/ (AnalyzeIndustry,start))     \\ initial(DetermineScope)
2 !(ManagementSurvey,start) W (DefiningMethodology,complete)     \\ precedence(
      DefiningMethodology,ManagementSurvey)
3 !(ProblemIdentificationWorkshop,start) W (DefiningMethodology,complete)     \\
      precedence(DefiningMethodology,ProblemIdentificationWorkshop)
4 !(AnalyzeIndustry,start) W (DefiningMethodology,complete)     \\ precedence(
      DefiningMethodology,AnalyzeIndustry)
5 (<>(ManagementSurvey,complete)=>!(<>(ProblemIdentificationWorkshop,complete)))
       /\ (<>(ProblemIdentificationWorkshop,complete)=>!(<>(ManagementSurvey,
      Complete)))     \\ exclusive choice(ManagementSurvey,
      ProblemIdentificationWorkshop)
```

Cross-Paradigm Transitions

While the impact of the same paradigm transitions on the desirable characteristics of a business process is rather limited, the impact of the cross-paradigm transitions can be rather extensive.

Imperative - Declarative Transition. The imperative process model is translated into a set of event-based business rules (e.g. preconditions), which can be used for a declarative process enactment (Casati et al. 1998). While the focus is mostly placed on the translation of the control-flow, Dumas et al. describe an approach to deal with the data-flow in UML activity diagrams (Dumas et al. 2005).

Impact on process characteristics. Since the implicit constraints governing the imperative process model are exactly mapped on event-based business rules, the *issue of overspecification is not dealt with*. However, several researchers have argued that the *process flexibility slightly increases* compared to the imperative-imperative transition due to the possibility of replacing or adding service task at run-time and ability to define extra event-based business rules at run-time to deal with temporary circumstances (Dumas et al. 2005).

The impact on the overall process compliance of this transition depends on the extent to which all the conditions of the imperative model can be modeled and translated into business rules. While the translation of a control-flow into business rules is well covered, the *translation of organizational and data conditions is not extensively described*. The process effectiveness and efficiency are determined by the quality of the process model and the declarative execution environments.

Example process case. The imperative-declarative transition is suitable for use within the context of *distributed processes*, for which the process environment remains relatively stable and the ability to dynamically deal with temporary circumstances (e.g. changing agents, service tasks, etc.) is valued. Typically these processes can be found in virtual organizations, consisting of long term alliances with a relative fixed structure that tolerates little variation in terms of partners. Examples are the complex order-to-cash processes in the automotive industry. In this section we elaborate the declarative run-time process model for the order-to-cash process in figure 2. Listing 6 provides the description of several process fragments for an event-based process deployment and execution infrastructure, e.g. the workflow management system in (Hens et al. 2014).

Listing 6: Imperative-Declarative Transition: Event Based Interaction of Process Fragments (base process figure 2, enaction prototype for this fragments in (Hens et al. 2014))

```
1 \\ Process fragment: Check Order and Client
2 Event rule: OrderReceived
3 End signal: (CheckOrderAndClient,complete)
4
5 \\ Process fragment: Check Availability
6 Event rule: (CheckOrderAndClient,complete) /\ CreditworthyClient /\
      DeliverableProduct
7 End signal: (CheckAvailability,complete)
8
9 \\ Process fragment: Place Order with Supplier
10 Event rule: (CheckAvailability,complete) /\ NoStockAvailable
11 End signal: (PlaceOrderWithSupplier,complete)
```

Declarative - Imperative Transition. Before run-time a systematic procedure is used for the construction of an optimal control-flow with reference to a particular characteristic. The systematic procedures for constructing an optimal control-flow from a declarative process model are closely related to artificial intelligence planning techniques (Maghraoui et al. 2006, Hendler et al. 1990, Ferreira and Ferreira 2005).

The artificial intelligence literature has delivered a significant amount of contributions towards the deployment of artificial intelligence planning. Several authors described the deployment of artificial intelligence planning for component-based applications, e.g. configuration of software systems (Arshad et al. 2007), change management (Keller et al. 2004), etc. Another noteworthy set of contributions has been presented by the intelligent manufacturing research. The process-graph method (Friedler et al. 1992, Brendel et al. 2000, Peters et al. 2002, Keller and Bryan 2000, Liu et al. 2004) has been used to generate

workflows that minimize the production cost (Tick et al. 2006), that minimize the waste (Hertwig et al. 2002), etc.

Impact on process characteristics. While the declarative process specifications provide *extensive design-time flexibility, run-time flexibility of this transition remains limited* to the flexibility offered by imperative enactment. However, the declarative process model in combination with a time-efficient planning algorithm, allows for a *rapid adoption of new compliance requirements*. Moreover, when the imperative workflow engine does not support any of the run-time flexibility enhancing techniques, compliance can be easily checked against the declarative process model.

The use of an artificial planning algorithm might positively affect both the *process efficiency and effectiveness*, since an optimization criterion needs to be specified. The generated optimal control-flow can minimize the execution time, the overall cost, etc. In addition, compared to declarative process enactment the end-user will be sufficiently guided and supported.

Example process case. The declarative-imperative transition is useful for processes that require *far-reaching redesigns* at regular intervals and/or for processes that benefit from an *optimization* with reference to a certain criterium. These processes, however, are at the same time relatively *stable in the periods between those redesign phases*. Additionally, compared to the declarative process enactment the end-user will be sufficiently *guided* and supported, which might be a requirement in certain environments. Certain (governmental) administrative processes (e.g. subsidy application processing) are subject to yearly changes due to changing budgets, policies, regulations, etc. Moreover, these processes would benefit from an optimization in terms of process efficiency and the number of required interactions with clients (i.e. citizens). To illustrate the transition between declarative and imperative process models we specify the run-time model for the best practice in gynecologic oncology treatment (figure 3 and listing 7), based on the most effective previous executions and known order constraints. The resulting clinical pathway can be used as a guideline.

Listing 7: Declarative-Imperative Transition: Optimal process model for straightforward cancer treatment 3

```
<process xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
    >
...
  <sequence>
    <invoke name="Consult" />
    <invoke name="Teletherapy" />
    <invoke name="FollowUpConsult" />
    <switch>
      <case condition="AcceptableForHyperthermia">
        <sequence>
          <invoke name="Hyperthermia" />
        <\sequence>
      </case>
      <case condition="NOTAcceptableForHyperthermia">
      </case>
    </switch>
    <invoke name="Brachytherapy"  />
  </sequence>
</process>
```

Hybrid - Imperative Transition. Within the context of this transition the focus is primarily placed on hybrid models of the second type, the process models that combine a full imperative specification with a set of business rules. Before run-time the imperative

reference model is *customized to the specific needs* of a particular case by applying the set of customizing business rules (Kumar and Yao 2009, Hallerbach et al. 2010b).

Impact on process characteristics. Due to the hybrid process model as a starting position, a neat approach to process variant management is provided. Compared to the process variant management approach introduced in the imperative-imperative transition, *maintenance of requirements is not needlessly complicated* since there is no duplication of the base process. However, the customization of the process model must be performed correctly and completely in order not to affect the process effectiveness.

Example process case. This type of hybrid - imperative transition will most likely be used for supporting a set of business processes that all only slightly differ from a specific reference process. We also expect the business processes to have predictable execution paths. An insurance company, for example, often has slightly differing policies depending on the specific insurance product (i.e. automobile, home, etc.). Other examples can be found in situations where an organization is trying to differentiate services depending on client type and/or client status.

Listing 8 specifies the imperative run-time process model for the materialized hybrid insurance model in figure 5. This imperative model is the optimal organization for claim handling in the context of fire-insurances with claims smaller than 1000.

Listing 8: Hybrid-Imperative Transition: Imperative run-time model for processing fire-insurance claims less than 1000 (figure 5)

```
<process xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
    >
...
  <sequence>
    <invoke name="ClaimsIntake" />
    <invoke name="EvaluateClaim" />
    <switch>
      <invoke name="PayAdvance" />
      <invoke name="ReviewDamages" />
      <invoke name="ProposeSettlement" />
      <invoke name="ApprovePayment" />
      <invoke name="Pay" />
      <invoke name="CloseClaim" />
    </switch>
  </sequence>
</process>
```

Translating declarative placeholders. Naturally, hybrid process models with placeholder activities can be transformed into imperative execution models as described in the declarative-imperative transition, which also results in a hybrid-imperative transition. Characteristics of the transition in this context are comparable with those of the declarative-imperative transition.

Conclusion

Designing information systems that provide support for operational business processes with the right level of process flexibility, compliance, efficiency and effectiveness can be a challenging task. This position paper promotes a clear distinction between the business process strategies and their differences at distinct points in the process life cycle. A clear overview of the different design-time and run-time positions was presented.

Furthermore, the paper elaborated on the transitions between design-time and run-time; in addition to the same paradigm transitions three interesting cross-paradigm transitions were presented.

Each of the presented business process positions and transitions has a different approach towards their focus in the business process management research, which results in differences in the extent to which the desirable characteristics are present in the resulting business process support. The optimal selection of design-time and run-time positions (and consequently the transition type) will be based on the business environment, see table 1. This paper presented detailed analyses of the different positions and transitions which might serve as a guide to determine the optimal selection of design-time and run-time positions.

Table 1.: Transition selection table

	Imperative Process Modeling	**Hybrid Process Modeling**	**Declarative Process Modeling**
Imperative Process Enactment	Processes with stable environment and predictable execution paths	Configuration of process variants that slightly differ from a stable reference model	Processes that: (a) requires regular far reaching redesigns with stability in the periods in between the redesigns *or* (b) Processes that can benefit from an optimization with reference to a certain criterion *or* (c) Processes that need guided support
Hybrid Process Enactment		Processes that consist of: (a) a combination of process paths in a stable and a highly evolving environment *and/or* (b) a combination of process parts with predictable and non-predictable execution paths	
Declarative Process Enactment	Distributed processes		Processes with (a) Highly evolving environment *and/or* (b) non-predictable execution paths

A shift from the narrow workflow transition focus towards an process transition focus, including all four perspectives, will create major opportunities for the research domain. Additionally, the design-time to run-time transition approach should be able to take into account a wide variety of optimization criteria, such as cost and time.

References

Adams, M., ter Hofstede, A., van der Aalst, W., Edmond, D., 2007. Dynamic, extensible and context-aware exception handling for workflows. On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS, 95–112.

Aguiar, M. W. C., Weston, R. H., 1995. A model-driven approach to enterprise integration. International Journal of Computer Integrated Manufacturing 8 (3), 210–224.

Arshad, N., Heimbigner, D., Wolf, A., 2007. Deployment and dynamic reconfiguration planning for distributed software systems. Software Quality Journal 15 (3), 265–281.

Bandara, W., Gable, G., Rosemann, M., 2005. Factors and measures of business process modelling: model building through a multiple case study. European Journal of Information Systems 14 (4), 347–360.

Barba, I., Del Valle, C., 2011. A planning and scheduling perspective for designing business processes from declarative specifications. In: Filipe, J., Fred, A. (Eds.), Proceedings of the 3rd International Conference on Agents and Artificial Intelligence. Vol. 1. Rome, Italy, pp. 562–569.

Barba, I., Weber, B., Del Valle, C., 2012. Supporting the optimized execution of business processes through recommendations. In: Business Process Management Workshops. Springer, pp. 135–140.

Bhat, J., Deshmukh, N., 2005. Methods for modeling flexibility in business processes. In: Workshop on Business Process Modeling, Design and Support (BPMDS05), Proceedings of CAiSE05 Workshops.

Brendel, M., Friedler, F., Fan, L., 2000. Combinatorial foundation for logical formulation in process network synthesis. Computers & Chemical Engineering 24 (8), 1859–1864.

Burstein, M., Bussler, C., Finin, T., Huhns, M., Paolucci, M., Sheth, A., Williams, S., Zaremba, M., 2005. A semantic web services architecture. Internet Computing, IEEE 9 (5), 72–81.

Bussler, C., 1996. Workflow-management-systems as enterprise engineering tools. In: Modelling and methodologies for enterprise integration. Springer, pp. 234–247.

Caron, F., Vanthienen, J., De Weerdt, J., Baesens, B., 2012. Advanced care-flow mining and analysis. In: et al., F. D. (Ed.), BPM 2011 Workshops, Part 1. Vol. 99 of LNBIP. Springer-Verlag Berlin Heidelberg, pp. 167–168.

Casati, F., Ceri, S., Pernici, B., Pozzi, G., 1998. Deriving active rules for workflow enactment. In: Database and Expert Systems Applications. Springer, pp. 94–115.

Chiao, C. M., Künzle, V., Reichert, M., 2013. Schema evolution in object and process-aware information systems: Issues and challenges. In: Business Process Management Workshops. Springer, pp. 328–339.

Chua, D. K., Nguyen, T., Yeoh, K., 2013. Automated construction sequencing and scheduling from functional requirements. Automation in Construction.

Curtis, B., Kellner, M. I., Over, J., 1992. Process modeling. Communications of the ACM 35 (9), 75–90.

Davenport, T., 1993. Process innovation: reengineering work through information technology. Harvard Business Press.

Decker, G., Dijkman, R., Dumas, M., García-Bañuelos, L., 2008. Transforming BPMN diagrams into YAWL nets. Business Process Management, 386–389.

Doehring, M., Zimmermann, B., Godehardt, E., 2010. Extended workflow flexibility using rule-based adaptation patterns with eventing semantics. In: Informatik2010 Service Science, Workshop: Business-Rule basierte Servicesteuerung. pp. 195–200.

Dumas, M., Fjellheim, T., Milliner, S., Vayssière, J., 2005. Event-Based Coordination of Process-Oriented Composite Applications. Business Process Management, 236–251.

Ellis, C., Nutt, G., 1993. Modeling and enactment of workflow systems. Application and Theory of Petri Nets 1993, 1–16.

ESPRIT Consortium AMICE, 1993. CIMOSA: Open Systems Architecture for CIM, 2nd revised and extended edition. Springer-Verlag, Berlin.

Fahland, D., Lubke, D., Mendling, J., Reijers, H., Weber, B., Weidlich, M., Zugal, S., 2009a. Declarative versus imperative process modeling languages: The issue of under-

standability. Enterprise, Business-Process and Information Systems Modeling, 353–366.

Fahland, D., Mendling, J., Reijers, H., Weber, B., Weidlich, M., Zugal, S., 2009b. Declarative vs. Imperative Process Modeling Languages: The Issue of Maintainability. In: 1st International Workshop on Empirical Research in Business Process Management. Ulm, Germany, pp. 65–76.

Fan, M., Stallaert, J., Whinston, A., 2000. The adoption and design methodologies of component-based enterprise systems. European journal of information systems 9 (1), 25–35.

Ferreira, D., Ferreira, H., 2005. Learning, planning, and the life cycle of workflow management. In: EDOC Enterprise Computing Conference, 2005 Ninth IEEE International. IEEE, pp. 39–45.

Fickas, S., 1989. Design issues in a rule-based system. Journal of Systems and Software 10 (2), 113–123.

Friedler, F., Tarjan, K., Huang, Y., Fan, L., 1992. Combinatorial algorithms for process synthesis. Computers & Chemical Engineering 16, S313–S320.

Ghose, A., Koliadis, G., 2007. Auditing business process compliance. Service-Oriented Computing–ICSOC 2007, 169–180.

Goedertier, S., Vanthienen, J., 2009. An overview of declarative process modeling principles and languages. Vol. 6. Communications of systemics and informatics world network, pp. 51–58.

Hallerbach, A., Bauer, T., Reichert, M., 2010a. Capturing variability in business process models: the provop approach. Journal of Software Maintenance and Evolution: Research and Practice 22 (6-7), 519–546.

Hallerbach, A., Bauer, T., Reichert, M., 2010b. Configuration and management of process variants. In: Handbook on Business Process Management 1. Springer Berlin Heidelberg, pp. 237–255.

Hendler, J., Tate, A., Drummond, M., 1990. AI planning: Systems and techniques. AI magazine 11 (2), 61.

Hens, P., Snoeck, M., Poels, G., Backer, M. D., 2014. Process fragmentation, distribution and execution using an event-based interaction scheme. Journal of Systems and Software.

Herbst, J., Karagiannis, D., 1999. An inductive approach to the acquisition and adaptation of workflow models. In: Proceedings of the IJCAI. Vol. 99. pp. 52–57.

Hertwig, T., Xu, A., Nagy, A., Pike, R., Hopper, J., Yaws, C., 2002. A prototype system for economic, environmental and sustainable optimization of a chemical complex. Clean Technologies and Environmental Policy 3 (4), 363–370.

Hrgovcic, V., Woitsch, R., Karagiannis, D., 2011. Hybrid service modeling in enterprise computing. In: Commerce and Enterprise Computing (CEC), 2011 IEEE 13th Conference on. IEEE, pp. 207–212.

Hur, W., Bae, H., Kang, S., 2003. Customizable workflow monitoring. Concurrent Engineering 11 (4), 313.

Kappel, G., Rausch-Schott, S., Retschitzegger, W., 1998. Coordination in workflow management systems a rule-based approach. Coordination Technology for Collaborative Applications, 99–119.

Karagiannis, D., Junginger, S., Strobl, R., 1996. Introduction to business process management systems concepts. In: Business process modelling. Springer, pp. 81–106.

Keller, A., Hellerstein, J., Wolf, J., Wu, K., Krishnan, V., 2004. The CHAMPS system: Change management with planning and scheduling. In: IEEE/IFIP Network Opera-

tions and Management Symposium NOMS 2004. IEEE Press.

Keller, G., Bryan, P., 2000. Process engineering: Moving in new directions. Chemical engineering progress 96 (1), 41–50.

Keller, G., Nuttgens, M., Scheer, A., 1992. Semantische Prozessmodellierung auf der Grundlage Ereignisgesteuerter Prozessketten (EPK). Inst. fur Wirtschaftsinformatik.

Kumar, A., Yao, W., 2009. Process Materialization Using Templates and Rules to Design Flexible Process Models. In: Proceedings of the 2009 International Symposium on Rule Interchange and Applications. Springer-Verlag, pp. 122–136.

Li, C., Reichert, M., Wombacher, A., 2008. On measuring process model similarity based on high-level change operations. Conceptual Modeling-ER 2008, 248–264.

Liu, J., Fan, L., Seib, P., Friedler, F., Bertok, B., 2004. Downstream process synthesis for biochemical production of butanol, ethanol, and acetone from grains: Generation of optimal and near-optimal flowsheets with conventional operating units. Biotechnology progress 20 (5), 1518–1527.

Lu, R., Sadiq, S., 2007. A survey of comparative business process modeling approaches. In: Business Information Systems. Springer, pp. 82–94.

Lu, R., Sadiq, S., Governatori, G., 2009. On managing business processes variants. Data & Knowledge Engineering 68 (7), 642–664.

Maghraoui, K., Meghranjani, A., Eilam, T., Kalantar, M., Konstantinou, A., 2006. Model driven provisioning: Bridging the gap between declarative object models and procedural provisioning tools. Middleware 2006, 404–423.

Martens, A., Lakshmanan, G., Mukhi, N., Khalaf, R., 2011. Integrated case management history and analytics. In: Data Engineering Workshops (ICDEW), 2011 IEEE 27th International Conference on. IEEE, pp. 238–242.

Mayer, R. J., Menzel, C. P., Painter, M. K., Dewitte, P. S., Blinn, T., Perakath, B., 1995. Information integration for concurrent engineering (IICE) IDEF3 process description capture method report. Tech. rep., Logistics Research Division, Wright-Patterson AFB.

Object Management Group, 2004. UML 2.0 superstructure specification. http://www.omg.org/spec/UML/2.0/.

Object Management Group, 2006. Business Process Modeling Notation (BPMN) 1.1. http://www.omg.org/spec/BPMN/1.1/.

Organization for the Advancement of Structured Information Standards (OASIS), 2007. Web services business process execution language 2.0. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf.

Österle, H., 1995. Business in the information age: Heading for new processes. Springer Verlag.

Ouyang, C., Dumas, M., Aalst, W., Hofstede, A., Mendling, J., 2009. From business process models to process-oriented software systems. ACM transactions on software engineering and methodology (TOSEM) 19 (1), 1–37.

Papazoglou, M., Georgakopoulos, D., 2003. Service-oriented computing. Communications of the ACM 46 (10), 25–28.

Paschke, A., Boley, H., 2009. Handbook of Research on Emerging Rule-Based Languages and Technologies: Open Solutions and Approaches. IGI Publishing, Ch. Rules capturing events and reactivity.

Pesic, M., Aalst, W., Eijnatten, F., 2008. Constraint-based workflow management systems: shifting control to users. Ph.D. thesis, Technische Universiteit Eindhoven.

Pesic, M., Schonenberg, H., van der Aalst, W., 2007. Declare: Full support for loosely-structured processes. In: Proceedings of the Eleventh IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007). Citeseer, pp. 287–298.

Pesic, M., van der Aalst, W., 2006. A declarative approach for flexible business processes management. In: Business Process Management Workshops. Springer, pp. 169–180.

Peters, M., Timmerhaus, K., West, R., Timmerhaus, K., West, R., 2002. Plant design and economics for chemical engineers. McGraw-Hill New York.

Qin, J., Fahringer, T., 2012. Automatic scientific workflow composition. In: Scientific Workflows. Springer, pp. 135–165.

Recker, J., 2010. Continued use of process modeling grammars: the impact of individual difference factors. European Journal of Information Systems 19 (1), 76–92.

Recker, J., Mendling, J., 2006. On the translation between bpmn and bpel: Conceptual mismatch between process modeling languages. In: The 18th International Conference on Advanced Information Systems Engineering. Proceedings of Workshops and Doctoral Consortium. pp. 521–532.

Regev, G., Soffer, P., Schmidt, R., 2006. Taxonomy of flexibility in business processes. In: Proceedings of the 7th Workshop on Business Process Modelling, Development and Support.

Reichert, M., Dadam, P., 1998. ADEPT flexsupporting dynamic changes of workflows without losing control. Journal of Intelligent Information Systems 10 (2), 93–129.

Reichert, M., Weber, B., 2012. Constraint-based process models. In: Enabling Flexibility in Process-Aware Information Systems. Springer, pp. 341–374.

Rosemann, M., van der Aalst, W., 2007. A configurable reference modelling language. Information Systems 32 (1), 1–23.

Sadiq, S., Orlowska, M., Sadiq, W., 2005. Specification and validation of process constraints for flexible workflows. Information Systems 30 (5), 349–378.

Sadiq, S., Sadiq, W., Orlowska, M., 2001. Pockets of flexibility in workflow specification. Conceptual ModelingER 2001, 513–526.

Scheer, A. W., 2000. ARIS: business process modeling. Springer.

Schmidt, K., Simonee, C., 1996. Coordination mechanisms: Towards a conceptual foundation of CSCW systems design. Computer Supported Cooperative Work (CSCW) 5 (2), 155–200.

Schmidt, R., 2006. Flexibility in service processes. In: Proceedings of the CAISE 2006 Workshop on Business Process Modelling, Development and Support, BPMDS.

Schonenberg, H., Mans, R., Russell, N., Mulyar, N., Aalst, W., 2008a. Process flexibility: A survey of contemporary approaches. Advances in Enterprise Engineering I, 16–30.

Schonenberg, H., Weber, B., Dongen, B., Aalst, W., 2008b. Supporting Flexible Processes through Recommendations Based on History. In: Proceedings of the 6th International Conference on Business Process Management. Springer-Verlag, p. 66.

Sinur, J., March 2009. The art and science of rules vs. process flows. Gartner Research, ID Number G00166408.

Swenson, K., Palmer, N., 2010. Mastering the Unpredictable: How Adaptive Case Management Will Revolutionize the Way That Knowledge Workers Get Things Done. Meghan-Kiffer Press.

Tick, J., Kovacs, Z., Friedler, F., 2006. Synthesis of optimal workflow structure. Journal of Universal Computer Science 12 (9), 1385–1392.

van der Aalst, W., 1997. Verification of workflow nets. Application and Theory of Petri Nets, 407–426.

van der Aalst, W., Adams, M., Hofstede, A., Pesic, M., Schonenberg, H., 2009a. Flexibility as a Service. In: Database Systems for Advanced Applications. Springer-Verlag, pp. 319–333.

van der Aalst, W., Pesic, M., 2006. DecSerFlow: Towards a truly declarative service flow

language. Web Services and Formal Methods, 1–23.

van der Aalst, W., Pesic, M., Schonenberg, H., 2009b. Declarative workflows: Balancing between flexibility and support. Computer Science-Research and Development 23 (2), 99–113.

van der Aalst, W., Ter Hofstede, A., 2005. YAWL: yet another workflow language. Information Systems 30 (4), 245–275.

van der Aalst, W., Weske, M., Grunbauer, D., 2005. Case handling: a new paradigm for business process support. Data & Knowledge Engineering 53 (2), 129–162.

Van der Aalst, W. M., 2011. Process mining. Springerverlag Berlin Heidelberg.

van der Aalst, W. M., Reijers, H. A., Weijters, A. J., van Dongen, B. F., Alves de Medeiros, A., Song, M., Verbeek, H., 2007. Business process mining: An industrial application. Information Systems 32 (5), 713–732.

Vemadat, F., 1998. Handbook on Architectures of Information Systems. Berlin: Springer-Verlag. pp. 243-264, Ch. The CIMOSA Languages.

Weber, B., Reichert, M., Rinderle-Ma, S., 2008. Change patterns and change support features-enhancing flexibility in process-aware information systems. Data & knowledge engineering 66 (3), 438–466.

Weber, B., Sadiq, S., Reichert, M., 2009. Beyond rigidity–dynamic process lifecycle support. Computer Science-Research and Development 23 (2), 47–65.

Weske, M., 2007. Business Process Management: Concepts, Languages, Architectures. Springer-Verlag New York Inc.

Yu, J., Manh, T., Han, J., Jin, Y., Han, Y., Wang, J., 2006. Pattern based property specification and verification for service composition. Web Information Systems–WISE 2006, 156–168.

Zeng, L., Flaxer, D., Chang, H., Jeng, J., 2002. PLM flowDynamic Business Process Composition and Execution by Rule Inference. Technologies for E-Services, 51–95.

Zisman, M., 1977. Representation, specification and automation of office procedures. Ph.D. thesis, Wharton School.

zur Muehlen, M., Indulska, M., Kamp, G., 2007. Business process and business rule modeling: A representational analysis. In: EDOC Conference Workshop, 2007. EDOC'07. Eleventh International IEEE. IEEE, pp. 189–196.